Automatic Sanskrit Poetry Classification Based on Kāvyaguṇa

Amruta Barbadikar and **Amba Kulkarni**
amruta.barbadikar@gmail.com, ambakulkarni@uohyd.ac.in
Department of Sanskrit Studies,
University of Hyderabad.

Abstract

Kāvyaguṇa denotes the syntactic and phonetic attributes or qualities of Sanskrit poetry that enhance its artistic appeal, commonly classified into three categories: Mādhyurya (Sweetness), Oja (Floridity), and Prasāda (Lucidity). This paper presents the Kāvyaguṇa Classifier, a machine learning module, designed to classify Sanskrit literary texts into three distinct guṇas, by employing a diverse range of machine learning algorithms, including Random Forest, Gradient Boosting, XGBoost, Multi-Layer Perceptron and Support Vector Machine. For vectorization, we employed two methods: the neural network-based `Word2vec` and a custom feature engineering approach grounded in the theoretical understanding of Kāvyaguṇas as described in Sanskrit poetics. The feature engineering model significantly outperformed, achieving an accuracy of up to 90.6% in K-fold cross-validation and 92% in Holdout validation.

1 Introduction

Natural language processing (NLP) involves the intersection of computer science, artificial intelligence, and computational linguistics, focusing on how computers can understand, interpret, and respond to human languages (Jurafsky and Martin, 2009). NLP is developing vastly for different languages with data-driven approaches. However, Low Resource Languages face numerous challenges in NLP, including data scarcity, complex morphology, orthographic and historical variations, and a lack of digital resources. These issues, combined with the inadequacy of models designed for other languages, hinder the development of robust NLP solutions for Sanskrit. Although Sanskrit has traditionally been categorized as a Low-resource language, its NLP development has gained significant momentum in recent years. This progress can be attributed to the well-defined linguistic theories and the association of modern computational approaches. Several computer applications are available for the analysis of the syntactic components of the language (Goyal and Huet, 2013; Satuluri and Kulkarni, 2013; Kulkarni and Shukl, 2009; Kulkarni and Kumar, 2013, 2011; Goyal et al., 2009; Kulkarni et al., 2020; Kulkarni and Das, 2012; S, 2022).

The poetic literature of a language reflects the higher intellect of society. Sanskrit literature, dating back to the Vedic era, has fascinated humanity with its sophisticated aesthetic dimensions. Classical Sanskrit literature is rich in examples of such poetic expression. Understanding and processing this literature require a high level of intellect. ĀNANDAVARDHANA emphasizes the importance of deeper understanding, stating, "Simply knowing the words and their meanings does not enable one to fully appreciate poetry; true enjoyment comes to those experts who grasp its deeper essence."[1] For machines lacking world knowledge, this becomes a complex task. Without advanced methods for the automatic interpretation of implied meanings, machines struggle to process the ornate language accurately.

While some progress has been made in the processing of literary regard, machines can more easily analyze aspects where meaning is not deeply embedded. For example, tasks such as meter identification (Rajagopalan, 2018; Melnad et al., 2015; Neil, 2023; Terdalkar and Bhattacharya, 2023), and the identification and analysis of sound figures like Yamaka (Barbadikar and Kulkarni, 2023) and Anuprāsa (Barbadikar and Kulkarni, 2024) have been successfully achieved.

Poetry classification is performed using various standards such as sentiment, poetic style, etc. Wujastyk (1978) worked out the classification for

[1] śabdārthaśāsanajñānamātreṇaiva na vedyate |
vedyate sa tu kāvyārthatattvajñaireva kevalam ‖ 1.7, Dhvanyāloka

Sanskrit classical text of *Kumārasambhava* based on meter scansion for authorship criteria. Ahmad et al. (2020) classified poetry text into emotional states using deep learning techniques. Deshmukh et al. (2021) performed sentimental classification of Marathi poems using machine learning. Singh et al. (2023) attempted classifying Hindi poetry on the phonemic features *komala* and *kathora* with the use of learning algorithms using statistical data. Kaur and Saini (1970) developed Punjabi poetry classifiers using different textual features.

In this paper, we present a novel approach to classifying poetry based on various features such as Lucidity, Floridity and sweetness known as the *Kāvyaguṇas* (henceforth referred to as guṇa) using various machine learning models. In addition, we discuss the application and comparison of two different methods of vectorization, viz. `Word2vec` and feature engineering.

2 Background

Sanskrit poetics is primarily divided into six different schools, viz. *rasa* (aesthetic flavor), *alaṅkāra* (figures of speech), *rīti* (style), *dhvani* (suggestion), *aucitya* (propriety), and *vakrokti* (obliqueness). Each of these is based on what they consider the most essential factor in poetry. Various elements contribute to the beauty of poetry, and these six schools emphasize the different aspects deemed the most important in the enjoyment of poetry.

Although *guṇa* (quality) is not considered a separate school, its role is vital in the enjoyment of poetry, as it catalyzes the experience of *rasa*, the aesthetic flavor. *Guṇa* literally means "quality" refers to syntactic and phonemic features in the literary text that enhances its overall aesthetic experience.

There are several opinions of theoreticians on the number and nature of Kāvyaguṇas. Mainly, there are three kinds of perspectives over the classification of the guṇas found in the tradition.

1. **Classification with Ten guṇas**

   BHARATA (1st century AD) introduced the concept of Kāvyaguṇa in Naṭyaśāstra. These ten Kāvyaguṇas are ***Śleṣa*** (Union of word and intended meaning), ***Prasāda*** (Perspicuity), ***Samatā*** (Uniformity), ***Samādhi*** (Concentration), ***Mādhurya*** (Sweetness), ***Oja*** (Grandeur), ***Padasaukumārya*** (Agreeableness), ***Arthavyakti*** (Directness of expression), ***Udāratā*** (Exaltedness), and ***Kānti*** (Polish).

   VĀMANA (6th century AD) standardised *rīti* concept. He bestowed the spotlight to *ritī* in his Kāvyālaṅkāra-sūtravṛtti (KS). According to him *rīti* (style) is the vital element of the Kāvya. The *rītis* are the style of poets, shown through the choice of the syllables which are the smallest part of a word and length of compound words. According to VĀMANA there are ten Śabdaguṇas and ten Arthaguṇas. These ten guṇas of each type have the same name but different attributes according to the category. These are instrumental in identifying the *rīti*. The features of these guṇas are similar to that of BHARATA. Poets like DAṆḌIN, BĀṆBHAṬṬA use complex combinations of syllables and long compounds. Three dominant *rītis* are *Gauḍī*, *Pāñcālī* and *Vaidarbhī*. *Gauḍī rīti* abounds in the qualities of *Oja* (floridity) and *Kānti* (polish).[2] *Pāñcālī* is endowed with the qualities of Mādhurya (sweetness) and Saukumārya (softness).[3] *Vaidarbhī* diction consists of all the ten guṇas in proportion.[4]

2. **Classification with Three guṇas**

   The classification consists of three guṇas, viz. ***Mādhurya*** (Sweetness), ***Oja*** (Floridity), and ***Prasāda*** (Lucidity)—was introduced by BHĀMAHA. In his Kāvyālaṃkāra (4th century AD), BHĀMAHA explains these three qualities in terms of their syntactic characteristics, although he does not explicitly label them as guṇas. Instead, these qualities represent the essential attributes that a poet must employ to express sweetness, floridity, and lucidity in their work.

   BHĀMAHA and others linked the concept of guṇas with *rasa* (aesthetic experience), Unlike theorists who regarded *rīti* (style) as the soul of poetry. According to this view, guṇas help enhance and prepare the reader's mind for the experience of *rasa*. This approach was later followed by eminent scholars like ĀNANDAVARDHANA and MAMMAṬA.

   MAMMAṬA (11th AD), while defining kāvya (poetry), emphasized the inclusion of guṇas as essential elements.[5] He not only advo-

[2] Ojaḥ kāntimati gaudiyā ‖ 2.12, KS
[3] Mādhurya Saukumāryopapannā pāñcālī ‖ 13, KS
[4] samagraguṇopetā vaidarbhī ǀ 2.11, KS
[5] tadadoṣau śabdārthau

cated the threefold classification of guṇas but also provided a rationale for not considering the ten types of guṇas outlined by earlier scholars. According to MAMMAṬA, *Śleṣa*, *Samādhi*, *Udāratā*, and *Prasāda*, as defined by VĀMANA, are subsumed under *Oja*. *Mādhurya* remains as it is, while VĀMANA's *Arthavyakti* corresponds to MAMMAṬA's *Prasāda*. VĀMANA's *Samatā* refers to uniformity in writing, which can sometimes be a flaw, as a poet needs to adapt their style to suit the mood of the events described in the kāvya. *Saukumārya* (Softness) is understood as the absence of harshness, and *Kānti* (Polish) is regarded as the absence of vulgarity.

3. **Other Classifications**

   KUNTAKA and BHOJA contributed with other prominent classification schemes. KUNTAKA identified guṇas as indicators of the three *mārgas* (paths of style): ***Saukumārya*** (softness), ***Prasāda*** (lucidity), ***Lāvaṇya*** (charm), and ***Abhijātya*** (nobility or delight). While the names of these guṇas remain the same, their characteristics vary across each *mārga*. Additionally, KUNTAKA emphasized the importance of *Aucitya* (propriety) and *Saubhāgya* (grace or delight), which are essential guṇas without which poetry would lose its appeal.

   BHOJA, who was a renowned critic in the 10th century AD, expanded on the concept of guṇas in his treatise Sarasvatīkaṇṭhābharaṇa. In this work, he detailed 24 śabdaguṇas (qualities of sound) and 24 arthaguṇas (qualities of meaning), offering an extensive critique of each aspect of poetic expression.

Given the clarity of the three guṇa classification, we adopted this scheme for our poetry classification model. These three guṇas viz. *Mādhurya* (sweetness), *Oja* (floridity), and *Prasāda* (lucidity) effectively encapsulate the essence of other classification systems, distilling multiple qualities into a simpler framework. *Mādhurya* is characterized by soft and sweet constructions with minimal compounds, while *Oja* involves harsher, more complex constructions with longer compounds. *Prasāda*, the neutral guṇa, is present in all types of constructions and across all *rasas*.

In the ten guṇa classification, many qualities are defined by the absence of faults, making it difficult to assign precise vector values for supervised machine learning tasks. In contrast, the three guṇa scheme clearly differentiates the types, reducing the likelihood of class confusion. The distinct features of *Mādhurya*, *Oja*, and *Prasāda* provide a solid foundation for classification, ensuring better clarity and separation in the output of the model. The detailed characteristics of these three guṇas are as follows.

1. **Mādhurya (Sweetness):** It invokes a sense of mollification and melting of the reader's mind. Commonly associated with *Śṛṅgāra* (erotic), *Karuṇa* (compassionate), and *Śānta* (peaceful) *rasas*. It is characterized by a pleasing construction using soft consonants and short or no compound words.

2. **Oja (Floridity):** It provides a sense of lustrous expansion and is effective in depicting *Vīra* (heroic), *Bībhatsa* (disgusting), and *Raudra* (furious) *rasas*. It is identified by the use of harsh consonant combinations and lengthy compounds.

3. **Prasāda (Lucidity):** This quality ensures that the meaning of the text is easily comprehensible upon hearing. It is present across all kinds of *Rasas* and constructions.[6] There is no consonant combination and length of compound words specified to identify Prasāda guṇa. However, the literary text where there is absence of Mādhurya and Oja, can be considered under this category.
   (The syntactic and phonetic features for Mādhurya and Oja are shown in the table 1)

3 Implementation and Algorithm

In the previous section, we discussed the characteristics of guṇas as defined in the treatises of Sanskrit poetics. For individuals, it can be challenging to focus on and recall these nuanced features, as a human mind often processes these poetic effects collectively, intuitively, to classify poetry into the respective guṇas. Despite clearly defined rules and boundaries, rule-based systems struggle to handle the inherent uncertainty and complexity of Sanskrit poetry. Moreover, there is currently no model capable of effectively processing such poetry without

saguṇau analaṅkṛtī punaḥ kvāpi | 1.1, Kāvyaprakāśa

[6] śrutimātreṇa śabdāttu yenārthapratyayo bhavet | sādhāraṇaḥ samagrāṇāṃ sa prasādo guṇo mataḥ ‖ 8.76, Kāvyaprakāśa

	Mādhurya	**Oja**
Consonants	all except *t, ṭ, d, ḍ, ś, ṣ*	*t, ṭ, d, ḍ, ś, ṣ*
Conjuncts	nasal C + C *r*/ṇ + short V	unaspirated C + aspirated C *r*/ṇ + long V *r* + C / C + *r* C + C
Compounds	No or of smaller length	lengthy
Construction	moderate complex	extreme complex

Table 1: Phonetic and Syntactic Features of guṇas
(Note: C = consonant, V = vowel)

supervision. Therefore, we chose to pursue a balanced approach using supervised machine learning with feature engineering.

3.1 Dataset

The learning dataset consists of 672 Sanskrit verses, equally distributed among the three guṇas. These verses are primarily sourced from classical Sanskrit literature, including works such as Kumārasambhava, Buddhacarita, Ṛtusaṃhāra, Amaruśataka, Manusmṛti, Śiśupālavadha, Daśakumāracarita, Rāmāyaṇa and Mahābhārata. To introduce more variety, selected hymns (stotras) like Śivatāṇḍava-stotra, Mahiṣāsura-mardinī-stotra, etc. have also been added to the dataset.

For classification purposes, we consider each verse as a unit of input. While the diction of a poet often remains consistent throughout a poem, the choice of a specific guṇa is influenced by the rasa (emotional essence) of the event being described. Consequently, 224 verses have been assigned to each of the three guṇas in the dataset, providing a balanced distribution for training the model.

These verses undergo simple steps of data normalization. We remove punctuation marks and *avagraha* ('ऽ'), a special symbol used to indicate the elided akāra or ākāra. Once the data is cleaned, it is ready for the feature extraction process. This ensures that the dataset is prepared for machine learning algorithms to process it efficiently and produce accurate classifications.

3.2 Vectorization

Machines do not inherently understand words or alphabets, they only process numbers. Therefore, it is essential to convert text into numerical values based on specific criteria. This numerical representation of text features is called a vector. Several automatic models are available for converting text into vectors, such as `Word2Vec`. `Word2Vec` is a neural network-based model that generates vector representations of words, capturing their semantic meanings and relationships by training on large datasets. It produces word embeddings, where similar words are mapped to nearby points in the vector space, enabling efficient similarity calculations and other linguistic tasks.

However, in case of Sanskrit, tokenization for vectorization becomes particularly challenging due to the phenomena of *sandhi* (phonetic joining of words) and *samāsa* (compound words). Additionally, the presence of polysemous words (words with multiple meanings) complicates the vectorization process while using models trained on for other languages. To address these challenges, manual feature extraction based on the clues provided by Sanskrit poetics is a more suitable approach.

Here, for the comparison purposes, we apply both the techniques of vectorization viz. `Word2vec` and feature engineering.

3.2.1 Custom Feature Engineering

The clues to identify guṇas are already provided in the poetic tradition. We define the eight different features. These features are as follows.

1. **Number of syllables:** This feature is a count of total number of syllables in a verse.

2. **Number of words:** This feature represents the proportion of the number of words in a verse to the total number of letters. The number of words and letters may vary depending on the meter. For instance, the *Anuṣṭup* meter has 32 *akṣaras*, while the *Śārdūlavikrīḍita* meter has 48 *akṣaras*. To maintain uniformity

across meters, we calculate the proportion of words to letters.

3. **Number of lengthy compound words:** This feature measures the number of long compound words. A compound deemed to be long if it has more than 9 syllables. The number is arrived from the heuristics of the data used for training. The count of such words is divided by the total number of words in the verse. Generally, verses with more compounds or *sandhi* have fewer independent words. Oja guṇa verses tend to have fewer words than Mādhurya verses, with Prasāda constructions having even more independent words.

4. **Number of Oja syllables and conjuncts:** In *Kāvyaprakāśa*, MAMMAṬA defines typical syllables and conjunct combinations of Oja[7] and Mādhurya. For Oja, we count:

   - All retroflex consonants except nasals (*ṭ, ṭh, ḍ, ḍh, ś, ṣ*).
   - *r* and *ṇ* followed by long vowels.
   - Unaspirated consonants paired with aspirated consonants from the same articulation group (e.g., *k+kh*, *g+gh*, *c+ch*, *j+jh*).
   - Any consonant combined with itself (e.g., *t+t*, *th+th*, *v+v*).

   These combinations are counted across the verse and divided by the total number of letters. The percentage is taken as the feature value.

5. **Number of conjunct consonants for Mādhurya:** For Mādhurya, MAMMAṬA provides the following guidelines[8]:

   - The combinations used for Oja should be avoided.
   - Count conjunct consonants where any nasal is combined with a consonant of the same articulation point (e.g., *ṇ+k, ñ+c, ṇ+ṭ, n+t, m+p*). This count is divided by the total number of letters, and the percentage is considered the feature value.

6. **Number of long vowels:** This feature represents the percentage of long vowels (*ā, ī, ū, e, ai, o, au*) out of the total number of vowels in the verse.

7. **Number of *r* and *ṇ* followed by short vowels:** This feature is the frequency of *r* and *ṇ* followed by short vowels, divided by the total number of letters in a verse.

8. **Number of unaspirated consonants:** This feature captures the percentage of unaspirated consonants (*k, g, c, j, ṭ, ḍ, p, b*) that occur in Mādhurya constructions, divided by the total number of letters in the verse.

These features are calculated for each verse of the dataset as an eight-dimensional vector.

3.2.2 `Word2vec`

To compare the performance of the mentioned feature engineering assisted model with a neural network-based word embedding model, we employed the `Word2vec` technique, which generates vectors for each word in the text based on its semantic meaning and relationships with neighboring words. However, in the context of guṇa classification, we are more focused on phonemes rather than the word meaning or their semantic relationships. Therefore, we split the text into akṣaras and generated vectors accordingly. We set the vector size to 500 and, since our classification is at the verse level, we calculated a sentence vector that is the average of the vectors for all akṣaras within each verse to represent the entire verse.

3.3 Learning Models

The extracted vectors are stored for each verse and are used to train the machine learning models. For training purposes, we split the dataset into training and test sets. Here, we have split the dataset into 80% training and 20% testing to ensure that the model generalizes well.

We consider multiple models that are developed to train the machine, namely,

- **Random Forest (RF)**
- **Gradient Boosting (GB)**
- **Support Vector Machine (SVM)**
- **Multi-layer Perceptron (MLP)**
- **Extra Trees (ET)**

[7] yoga ādyatṛtīyāmyāmantyayo reṇa tulyayoḥ |
ṭādiḥ śaṣau vṛttidairghyaṃ gumpha uddhṛta ojasi ‖ 8.75, Kāvyaprakāśa

[8] murdhni vargāntyagāḥ sparśā aṭavargā raṇau laghū |
āvṛttirmadhyavṛttirvā mādhurye ghaṭanā tathā ‖ 8.74, Kāvyaprakāśa

- **Extreme Gradient Boosting (XGB)**

We optimise the performance of MLP by setting `max-iter = 2 000` and in SVM we set the `gamma = 'auto'`.

4 Performance Evaluation

4.1 Validation of `Word2vec`

The six models were then trained using these automatically generated vectors. The precision and standard deviation of these models evaluated using K-fold cross-validation are detailed in the table 2. We observed a low score in all the models. The highest score was obtained in the Multi-Layer Perceptron (MLP) at 59%.

Model	Accuracy	Standard Deviation
RF	0.56	0.06
GB	0.58	0.07
SVM	0.33	0.02
MLP	**0.59**	**0.11**
ET	0.56	0.06
XGB	0.58	0.06

Table 2: Performance Metrics in K-fold Validation of Different Models using `Word2vec`

4.2 Validation of Feature Engineering Based Models

To thoroughly test models trained on the custom features given by the poetic theorists, we considered two types of validation techniques.

4.2.1 10-fold Cross-validation

Model	Accuracy	Standard Deviation
RF	0.885106	0.031844
GB	0.870213	0.032197
SVM	**0.906383**	**0.043811**
MLP	0.893617	0.026913
ET	0.893617	0.028546
XGB	0.893617	0.026913

Table 3: Accuracy and Standard deviation scores of different models according 10-fold cross validation

The table 3 shows the accuracy and standard deviation of various machine learning models. The figure 1 shows the comparison of models in the box and whisker plot. The Support Vector Machine model achieved the highest accuracy at 90.6%, though with a slightly higher standard deviation (0.0438). Models like Gradient Boosting, Multi-Layer Perceptron, Extra Trees, Random Forest and XGBoosting all performed well, with accuracies around 87-89%, and comparable stability. The standard deviations for all models are relatively low, suggesting consistent performance across different runs, although the SVM model displayed slightly greater variability than the others. Overall, SVM stands out in accuracy, while other models offer a balanced mix of performance and consistency.

4.2.2 Holdout Validation

The other technique we employ to evaluate the model is the holdout validation technique, in which the dataset is split into two subsets. One for training the model and the other for testing its performance. This ensures that the model is evaluated on unseen data, providing a more reliable and generalized model when run on unseen data. There are various scores to compare the performances (see table 4). These scores cover various conditions for the reliability of the models. In addition, a confusion matrix is also generated to analyze the learning of the model for each of the classes.

Model	Accuracy
RF	0.88
GB	0.87
SVM	0.90
MLP	0.87
ET	**0.92**
XGB	0.91

Table 4: Performance Comparison of Machine Learning Models According to Holdout Validation

In the context of multi-class classification, a confusion matrix provides a detailed breakdown of the model's performance by showing how predictions for each class correspond to the true labels. It gives insight into how often the model confuses one class with another, highlighting strengths and weaknesses in classification accuracy across different classes. We reserved 20% of the data in the test set. 135 verses in the test set were randomly distributed among these classes as, 45 for Oja, 47 for Mādhurya and 43 for Prasāda.

In the matrices given for each model in tables 5, 6, 7, 8, 9 and 10 the rows indicate gold values for Mādhurya (M), Prasāda (P) and Oja (O). The columns M, P and O show the predicted classes.

The confusion matrices indicate that all the models were able to clearly distinguish between the

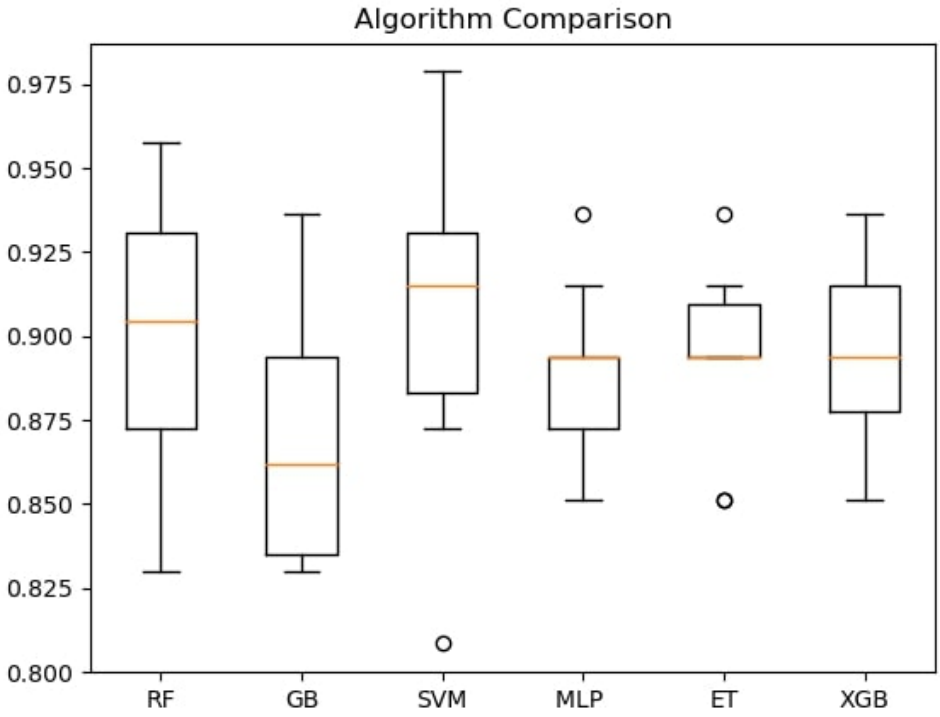

Figure 1: Comparison of Model Performance

	M	P	O
M	37	4	4
P	3	44	0
O	5	0	38

Table 5: **Random Forest**

	M	P	O
M	36	5	4
P	4	43	0
O	4	0	39

Table 6: **Gradient Boosting**

	M	P	O
M	38	4	3
P	3	44	0
O	4	0	39

Table 7: **Support Vector Machine**

	M	P	O
M	37	6	2
P	4	43	0
O	6	0	37

Table 8: **Multi-layer Perceptron**

	M	P	O
M	39	4	2
P	2	45	0
O	3	0	40

Table 9: **Extra Trees**

	M	P	O
M	38	4	3
P	2	45	0
O	3	0	40

Table 10: **XGBoost**

classes Oja (O) and Prasāda (P). Notably, there are no instances of O being falsely predicted as P or P being falsely predicted as O, demonstrating the models' strong ability to differentiate between these two guṇas. However, the models encountered difficulty in distinguishing Mādhurya (M) from both O and P. Specifically, the models misclassified 5–9 examples between M and P, while a similar range of misclassifications (5–9 instances) was observed between M and O. This fact highlights the

greater challenge in accurately distinguishing between M and other two classes.

5 Error Analysis

As we have seen in table 1, the characteristics of Mādhurya and Oja are contradictory to each other. Focusing on it we described various features. With this information, one would expect the models to perform better for Oja and Mādhurya on first place. Contrary to our expectation, we find that some Mādhurya examples are incorrectly classified as Oja and Prasāda and vice versa. In order to analyze the failure cases, we take the average of the values for each feature in different classes, and compare the values of the incorrectly analysed cases. The verse with the feature values (Table 12) for the following four cases are as follows.

Feature	M	P	O
Syllables (**v**)	126.76	80.98	137.66
Words (**w**)	10.90	13.13	6.27
Lengthy Compounds (**lc**)	4.72	0.98	28.58
Oja Syllables (**os**)	18.05	14.64	23.71
Long Vowels (**lv**)	17.46	20.90	17.19
Unaspirated Consonants (**uc**)	34.51	33.90	33.80
Mādhurya Syllables (**ms**)	8.73	7.79	6.66
r and *n* with short vowels (**rn**)	5.62	4.67	6.99

Table 11: Average value for each variable for categories M, P and O

1. **Mādhurya predicted as Oja:** In Table 12, the first example shows a verse, presented below, from the Oja class incorrectly classified as Mādhurya. The values of key variables such as **v**, **w**, **lc**, **os**, and **rn** closely align with the average values for the Mādhurya class, leading to the misclassification.

   dharasyoddhartāsi tvamiti nanu sarvatra jagati
   pratītastatkiṃ māmatibharamadhaḥ prāpipāyiṣuḥ |
   upālabdhevoccairgiripatiriti śrīpatimasau balākrāntaḥ
   kriḍaddviradamathitorvīruharavaiḥ||

2. **Oja predicted as Mādhurya:**
   In Table 12, the second example demonstrates a verse, shown below, from the Mādhurya class being incorrectly classified as Oja. The variables **v**, **w**, **lc**, **os**, and **rn** exhibit values that match the average characteristics of the Oja class, which likely contributed to the error.

   śrīmadbhirjitapulināni
   mādhavīnāmārohairnibiḍabṛhannitambabimbaiḥ|
   pāṣāṇaskhalanavilolamāśu nūnaṃ
   vailakṣyādyayuravarodhanāni sindhoḥ ||

3. **Mādhurya predicted as Prasāda:**
   The third instance in Table 12 represents a verse, given below, belonging to the Mādhurya class that was misclassified as Prasāda. Here, the values of the variables **v**, **w**, **lv**, and **ms** align more closely with the average values for Prasāda, resulting in the misclassification.

   tadīyamālokya sugāḍha bhakte
   mahāntamāveśamuvāca lokaḥ |
   baddhādarosau yadi pāṇḍuraṅge tadantike
   tiṣṭhati kiṃ na nityam ||

4. **Prasāda predicted as Mādhurya:**
   The fourth instance in Table 12 shows a verse, presented below, from the Prasāda class being classified as Mādhurya. This misclassification can be attributed to the variables **v**, **w**, and **lv**, whose values align closely with the average values of the Mādhurya class.

   acyutaṃ keśavaṃ rāmanārāyaṇaṃ
   kṛṣṇadāmodaraṃ vāsudevaṃ harim |
   śrīdharaṃ mādhavaṃ gopikāvallabham
   jānakīnāyakaṃ rāmacandraṃ bhaje ||

Oja is characterized by its complexity, while Prasāda is associated with simplicity. In contrast, Mādhurya, with its soft combinations of consonants and a moderate level of complexity, occupies a midpoint between these two guṇas. Even for a human annotator, it is tricky to classify the ślokas, especially those identified as failed cases, into a definitive guṇa. This observation provides a plausible explanation for the misclassifications, as texts exhibiting Mādhurya may sometimes share features with Oja and Prasāda.

This discrepancy opens avenues for further exploration. Future research could address this issue by employing multilabel classification techniques, allowing texts to be assigned to one or more guṇas based on their dominant features.

In Table 11, features such as **v**, **w**, **lc**, and **os** have shown a significant impact on the classification task. However, features like **lv**, **uc**, **ms**, and **rn** exhibit less pronounced differences in their average values across guṇas. Assigning weights to these features based on their relevance to specific classes could improve classification results. For instance, greater weight could be assigned to **lv**, **uc**, **ms**, and **rn** when they appear in Mādhurya.

Sr. No.	v	w	lc	os	lv	uc	ms	rn	True class	Predicted
1	183.0	6.5574	16.6667	34.0000	14.8649	34.4262	12.0219	6.0109	M	O
2	121.0	9.9174	0.0000	18.0000	17.6471	38.8430	4.9587	4.1322	O	M
3	99.0	13.1313	0.0000	13.0303	22.7273	37.3737	5.0505	1.0101	M	P
4	112.0	10.7143	0.0000	15.0000	14.8936	26.7857	10.7143	9.8214	P	M

Table 12: Misclassified examples with their feature values and predicted labels.

Such approaches could offer deeper insights into the overlapping characteristics of these poetic styles and enhance the accuracy of classification tasks.

6 Comparision of Feature Engineering with Word2vec

Word2vec is a widely used unsupervised learning technique that captures semantic relationships between words by mapping them into continuous vector spaces. However, in our task of classifying Sanskrit poetry into guṇas, we observed certain limitations when relying solely on Word2Vec embeddings.

The low score with the Word2vec model can be attributed to the limited size of the training data. Models like Word2vec typically require large datasets, often in the thousands or even millions of examples, to perform effectively. Additionally, the vector sizes used in such models usually range from hundreds to thousands, demanding powerful computational resources. Training on smaller datasets with limited resources poses significant challenges. Despite these constraints, our approach of training the model on minimal data with available resources has produced commendable results, highlighting the effectiveness of the feature engineering technique.

The custom feature engineering method is specifically tailored to the nuances of Sanskrit poetics. By incorporating features which indirectly reflect morphological structure, consideration of metrical patterns, and stylistic markers, our model was able to better capture the inherent characteristics that define different guṇas. This domain-specific approach resulted in higher accuracy compared to the more general-purpose Word2Vec embeddings.

7 Conclusion

The computational processing of poetic language presents significant challenges. To address these complexities, we introduced the module 'Kāvyaguṇa Classifier', designed to classify literary texts into three distinct guṇas: Mādhurya, Oja, and Prasāda. For this task, we explain two different approaches used for vectorization. Word2Vec is useful for capturing semantic information, our custom feature engineering leveraged expert knowledge of the literary domain, allowing a more precise classification in this context. This demonstrates that, in specialized tasks like the classification of Sanskrit poetry, combining traditional machine learning techniques with domain-specific insights can offer significant advantages over purely data-driven methods.

This module not only serves its primary purpose, but can also be adapted for related tasks. As discussed previously, these guṇas are essential components of both *rītis* and *rasas*. The *rītis* reflect the stylistic inheritance of poets, while the rasas encapsulate the emotions arising from aesthetic experiences. The interplay of guṇas in these contexts is crucial, making the Kāvyaguṇa classifier a valuable tool for identifying both *rītis* and *rasas*.

Moreover, given the linguistic similarities between Sanskrit and other Indian languages, their grammatical structures and poetic traditions are deeply influenced by Sanskrit. Concepts analogous to Kāvyaguṇa can be found in these languages as well. By employing language-specific training data and adapting feature extraction methodologies accordingly, similar classification tasks can be effectively achieved across various Indian languages.

8 Limitations

In this paper, we have discussed the models trained on engineered features limited to Sanskrit. To extend this method for other languages, similar exercise needs to be worked out by observing the language-specific poetic style.

References

Ācārya Viśveśvara. 2017. *Kāvyaprakāśa*. jñānamaṇḍala Limited, Varanasi.

Prateek Agrawal and Vishu Madaan. 2020. A Sanskrit to Hindi language machine translator using rule based approach. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON): System Demonstrations*, pages 13–15, Patna, India. NLP Association of India (NLPAI).

Shakeel Ahmad, Mohd Zubair, Fahad Alotaibi, and Sherafzal Khan. 2020. Classification of poetry text into the emotional states using deep learning technique. *IEEE Access*, PP:1–1.

Amruta Barbadikar and Amba Kulkarni. 2023. Yamaka identifier and classifier: A computational tool for the analysis of Sanskrit figure of sound (upcoming). In *Annals of Bhandarkar Oriental Research Institute*.

Amruta Barbadikar and Amba Kulkarni. 2024. Anuprāsa identifier and classifier: A computational tool to analyze Sanskrit figure of sound. In *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, pages 102–112, Auroville, Puducherry, India. Association for Computational Linguistics.

Rushali Deshmukh, Suraj Kore, Namrata Chavan, Sayali Gole, and Kumar Adarsh. 2021. Marathi poem classification using machine learning. *International Journal of Recent Technology and Engineering*, 8.

Benjamin Englard. 2013. A rhetorical analysis approach to natural language processing. In *ArXiv*.

Manomohan Ghosh. 1951. *The Nāṭyaśāstra*, volume 1. Asiatic Society of Bengal, Calcutta.

Pawan Goyal, Vipul Arora, and Laxmidhar Behera. 2009. Analysis of sanskrit text: Parsing and semantic relations. In *Sanskrit Computational Linguistics*, pages 200–218, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a sanskrit reader. In *Proceedings, 5th International Symposium on Sanskrit Computational Linguistics. DK Printworld (P) Ltd.*

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd edition. Pearson Prentice Hall.

Jasleen Kaur and Jatinderkumar Saini. 1970. Designing punjabi poetry classifiers using machine learning and different textual features. *The International Arab Journal of Information Technology (IAJIT)*, 17(01):38 – 44.

Amba Kulkarni and Monali Das. 2012. Discourse analysis of Sanskrit texts. In *Proceedings of the Workshop on Advances in Discourse Analysis and its Computational Aspects*, pages 1–16, Mumbai, India. The COLING 2012 Organizing Committee.

Amba Kulkarni and Anil Kumar. 2011. Statistical constituency parser for Sanskrit compounds. In *ICON 2011*.

Amba Kulkarni and Anil Kumar. 2013. Clues from Aṣṭādhyāyī for compound type identification. In *5th international SCLS 2013*.

Amba Kulkarni and Madhusoodana Pai. 2019. Sanskrit sentence generator. In *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 1–13, IIT Kharagpur, India. Association for Computational Linguistics.

Amba Kulkarni, Pavankumar Satuluri, Sanjeev Panchal, Malay Maity, and Amruta Malvade. 2020. Dependency relations for Sanskrit parsing and treebank. In *Proceedings of the 19th International Workshop on Treebanks and Linguistic Theories*, pages 135–150, Düsseldorf, Germany. Association for Computational Linguistics.

Amba Kulkarni and Devanand Shukl. 2009. Sanskrit morphological analyser: Some issues. In *the Festscrift volume of Bh. Krishnamoorty, Indian Linguistics*.

P. C. Lahiri. 1937. *Concepts of Rīti and Guṇa in Sanskrit Poetics*. Phd thesis, University of Dacca.

Keshav Melnad, Peter Scharf, and Pawan Goyal. 2015. Meter identification of Sanskrit verse. In *Sanskrit Syntax: Selected Papers Presented at the Seminar on Sanskrit Syntax and Discourse Structures*.

Dr. Kameshvar Nath Mishra. 2006. *Sarasvatīkaṇṭhābharaṇam by Bhoja*. Chaukhambha Publishers.

Radheshyam Mishra, editor. 2012. *Vakrokti-jīvita*. Chowkhamba Sanskrit Series, Benares.

Tyler Neil. 2023. Skrutable: Another step toward effective Sanskrit meter identification. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*.

S. Rajagopalan. 2018. A user-friendly tool for metrical analysis of Sanskrit verse. In *Computational Sanskrit & Digital Humanities, Selected papers presented at the 17th World Sanskrit Conference*.

Prasanna S. 2022. Spellchecker for Sanskrit:the road less taken. In *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*, pages 290–299, New Delhi, India. Association for Computational Linguistics.

C. Shankara Rama Sastri. 1956. *Kāvyālaṅkāra of Bhāmaha*. The Sri Balamanorama Press, Mylapore, Madras.

Pavankumar Satuluri and Amba Kulkarni. 2013. Generation of sanskrit compounds. In *ICON 2013*.

Shrikrishnamoori. 1909. *Kāvyālaṅkārasūtravṛttiḥ*. Sri Vani Vilas Press, Srirangam.

Ekaterina V. Shutova. 2011. *Computational approaches to figurative language*. Phd thesis, University of Cambridge.

Niraj Kumar Singh, Komal Naaz, and Soubhik Chakraborty. 2023. Komala and kathora: A novel approach towards classification of hindi poetry. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 22(6).

Krishnan Sriram, Amba Kulkarni, and Gérard Huet. 2023. Validation and normalization of DCS corpus and development of the Sanskrit heritage engine's segmenter. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 38–58, Canberra, Australia (Online mode). Association for Computational Linguistics.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2023. Chandojnanam: A Sanskrit meter identification and utilization system. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*.

Can Wang. 2022. Analysis of poetry style based on text classification algorithm. *Scientific Programming*, 2022.

Dominik Wujastyk. 1978. Automatic scansion of sanskrit poetry for authorship criteria. *Bulletin - Association of Literary and Linguistic Computing*, 6(2):122–135.